CHESS AI
USER GUIDE

PREPARED FOR

COSC3P71

PREPARED BY

Ralph Pereira - 4554879

Sammi Mak - 5931464

Contents

1. Introduction 2

2. Getting Started 2

2.1 Parameters 2

2.2 Game Mode 2

2.3 Board State 3

3. Making Moves 4

3.1 Move Processing 4

3.2 Player Movement 4

3.3 AI Movement 5

3.3.1 Heuristic/Evaluation Function 5

4. Events 6

4.1 En Passant. 6

4.2 Pawn Promotion 6

4.3 Castling 7

4.4 Checkmate 8

4.5 Winning the Game 8

5. Design 9

5.1 GUI 9

5.2 Pieces 9

5.3 AI 9
.

6. References 9

1. Introduction

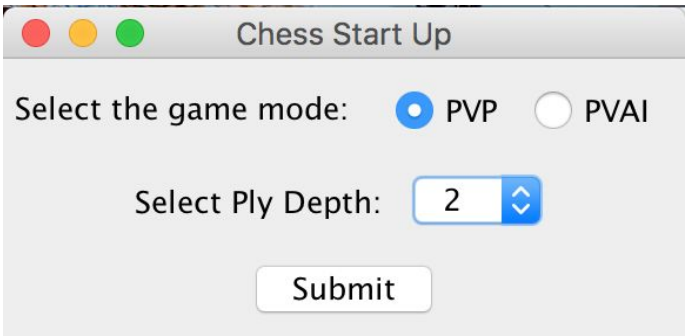
The purpose of this project is to apply topics learned from class in order to build a Chess engine with AI capability. The application will provide two game methods one where the player can play against the AI or without. The board will be represented as a classic 8x8 board with graphical representation of each piece. The game will follow all regular chess rules as well as special moves: pawn promotion and castling.

---

2. Getting Started

Primary Menu

Once the program is started there will be a start menu that the user must specify whether or not they want the AI playing in the game as well as the ply depth. The ply depth can be set to a value from 1 to 6 for testing purposes but is only reasonably functional up to a ply depth of 10 which takes about 5-10 seconds for the AI to make a move.



Game Modes

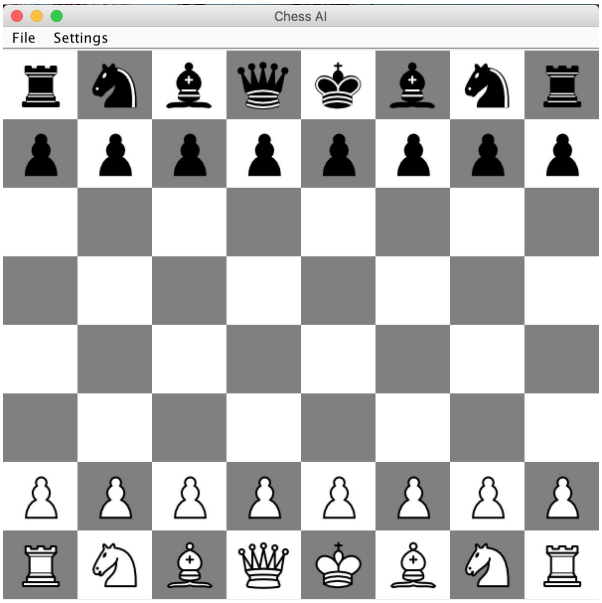
Upon starting the game, the user will always start on the white side with priority and either the AI or the second user/player will be on the black side.

Ply Depth

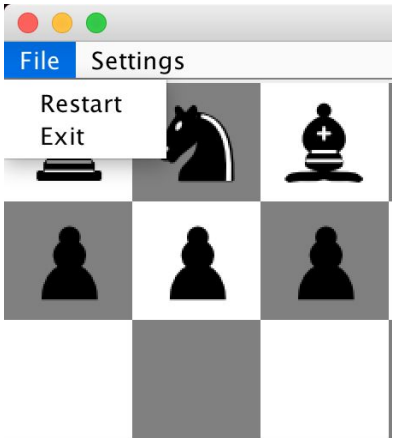
This value the user will specify is used for the search depth of the tree during the alpha beta pruning process. This is will limit how many turns into the the game will be examined into the future. There is a noticeable difference in logic between depth searches. The AI makes smart decisions past a depth of 4.

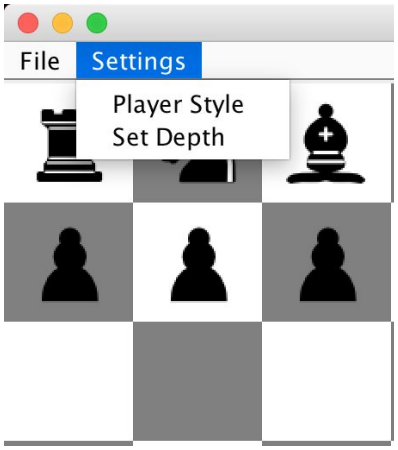
Board State

After inputting the initial parameters the beginning board state will be displayed with the user being the White side and always starting first.



There are also two menu options in the JMenuPane, one that was intended to allow the user to set ply depth during the middle of a game and change player style (to being White or Black). These features were omitted as the game functionality took higher priority than these GUI options. However, the restart option that works to reset the board and the exit option under File are both functional.





Making Moves

Move Processing

Once a player or AI moves a piece on the board, several cases must be examined:

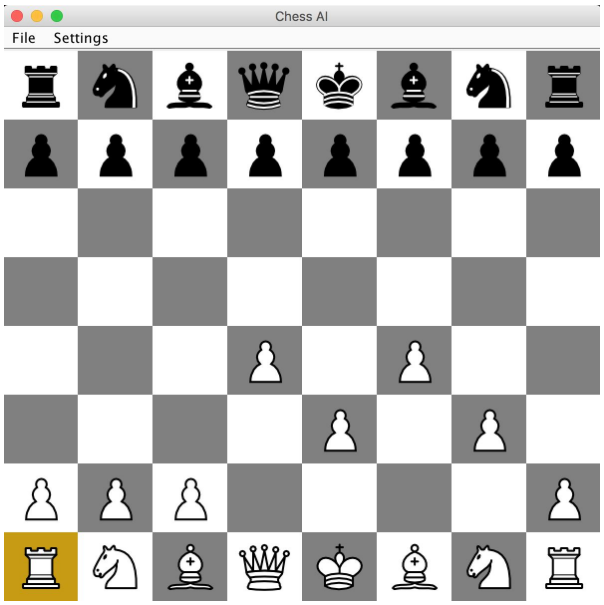
I. If the move was a castle
II. If the move was en passant
III. If a pawn can be promoted
IV. If the piece can attack a king
V. If a king is in check
VI. If the board is in checkmate/win state
VII. If a move was made can it attack the king

These checks are useful in predicting whether a move is better than another and can also be used to trigger a special event prompt.

Player Movement

The user will click to select the piece they want to move and the available moves will be highlighted in yellow. To place the piece to the desired location, the user will click on a highlighted location that will appear after the initial click.

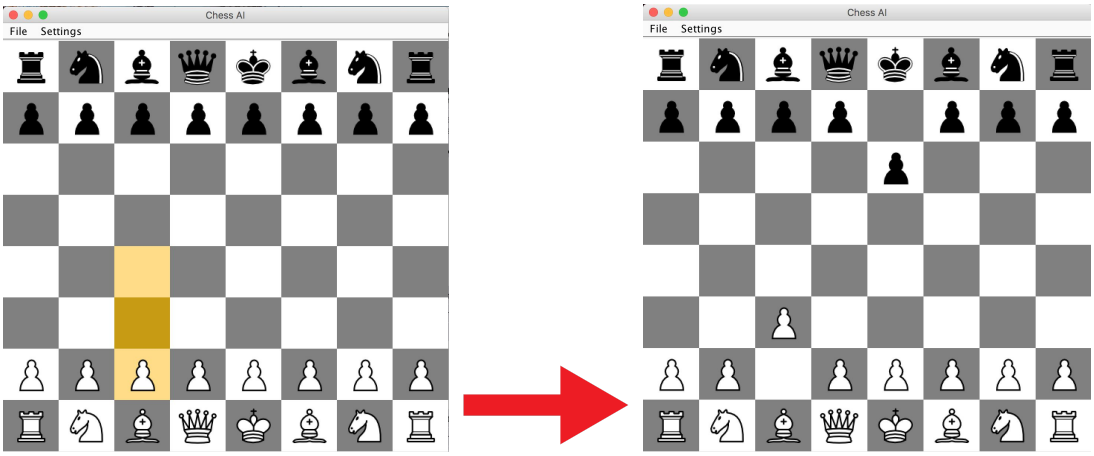
The image above depicts the board with the knight selected, with it's legal moves highlighted. If the user wants to choose a different piece to play they must first un-select the currently selected piece.



If there are no moves possible with a piece such as the rook above, only the piece itself will be highlighted and it will not be able to progress.

AI

With a reasonably ply depth, the AI movement will occur immediately after a user successfully moves a piece.



Evaluation Function

The program utilizes a simple symmetric evaluation function that was formulated in 1949 by Claude Shannon, a mathematician from MIT. This returns us an evaluation of the board state, or positional evaluation of the board as a result of a move. In our program, every piece contains a value that follows a hierarchy where

$$K > Q > R > B = N > P$$

The function is then to sum up the differences of each side's respective pieces multiplied by their own set values like so:

f(p) = 200(K-K')
+ 9(Q-Q')
+ 5(R-R')
+ 3(B-B' + N-N')
+ 1(P-P')

Where the opponent's piece is denoted by a prime.

Special Events

The following are the special moves that the game can play in order to gain a better winning advantage.

En Passant

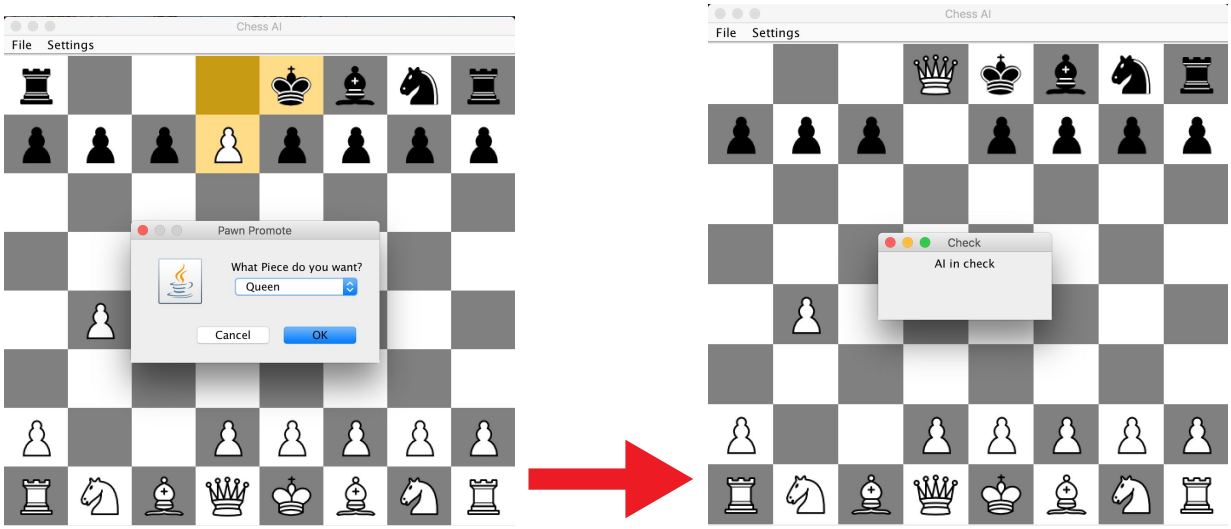
This is a move related to the pawn and happens immediately after a player's pawn makes a double step move as its first move where it was previously in a position to be captured by an opponent pawn in one step. This then allows on the next turn for the opponent's pawn to "pass through" to the location the player's pawn would have been had they taken a single step instead of a double, capturing their pawn in the process.

In our Pawn class we have a boolean firstMove to determine whether or not the pawn is still in it's starting state. From the ChessBoard class there is a method moveSpecialPiece() which sets the boolean to the pawn to false if it has already made its double step move. However, we did not plan enough time to fully implement this function.

Pawn Promotion

After a move is made, there is a method checkPromo() that checks if a pawn can be promoted if it is on the opponent's front row. There is a prompt that will ask the user whether they want their pawn to be replaced with:

I. Queen
II. Knight
III. Rook
IV. Bishop



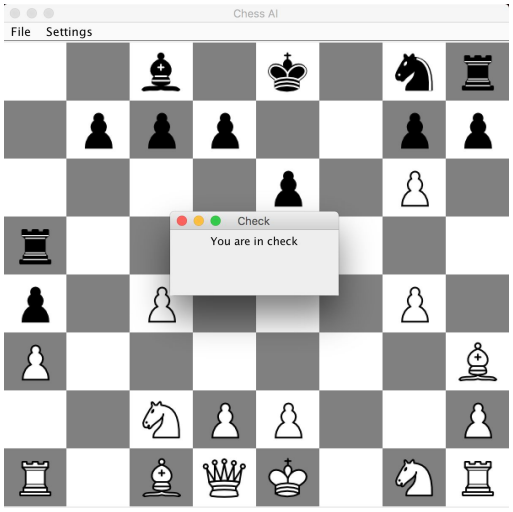
When selected, the game will replace that pawn piece on the board with the chosen piece at the same location. In the image above, it coincidentally places the AI in check as well.

Castling

This is another special move that allows a player to effectively protect the King from attack and usually occurs later in the game when most of the pieces have moved from its original positions. Castling is the act of swapping places between the Rook and King given the King is not in check and there are no pieces in between the two pieces. In the example below the King is selected and the rook is highlighted as a possible move to complete the castling move.

Checkmate

This is the end game for a game of chess, the player wants to put the opponent into checkmate. The program first checks to see if the king is in check, if the king is in check it then checks the number of moves left for the king. Since the king's moves are calculated with the king under attack in mind, if the number of moves left for the king are none, it's a checkmate.



Winning the Game

Once the game has verified that the King is in checkmate, the following prompt will appear to indicate that the player or AI has won the game.

Design

This section will discuss how parts of the program are implemented that were significant to us and may need further explanation.

GUI

Board

The board was created with three components in the project, GameTile, GameBoard, and GameFrame. The GameTile are all JPanels that delegate colors to make up the background of the board. This is also where the board calls it's updateTile() function when repainting a new state of the board. The GameBoard is where the game is processed, with the tileClickHandler() responds to the user's actions. This is also where functions such as the Pawn Promotion dialog is created.
All of this is then added into the GameFrame, along with the creation of the JMenuBar.

Pieces

The Pieces are implemented as an abstract class with every piece holding a PieceType, ColorType to delegate which piece is on which side. In every piece, logic is included for which moves are legal and has a getMoves() method to return a list of legal moves in relation to a position on the current state of the board.

AI

It was decided to keep the AI isolated in one class for abstraction purposes. During the alpha beta pruning process we decide to copy the board each time a new configuration is created during the search. This may be computationally more expensive but allows us to keep an isolated version of the current board without having to undo any moves the AI could make.

References

Isenberg, Gerd "Evaluation." https://chessprogramming.wikispaces.com/Evaluation